Программирование ЦАП модуля E14-140-MD

Приложение к руководству программиста.

1. Общие замечания

1.1. Отличия от версий Lusbapi 3.3, 3.4

Данное дополнение к руководству программиста E14-140-M подробно описывает возможности вывода на ЦАП, включая расширенные функции, доступные начиная с версии 3.10 встроенного ПО модуля.

Необходимо отметить, что актуальные на момент написания этого текста версии библиотеки Lusbapi 3.3, 3.4 соответствуют функциональности версии встроенного ПО 3.05 и не содержат всех структур и функций, описанных ниже.

Недостающая функциональность может быть реализована в прикладной программе с использованием недокументированных функций GetArray() и PutArray(). Необходимые примечания даны в разделе "*Функции для работы с ЦАП*".

Если же расширенные возможности версии 3.10 не требуются, можно применять Lusbapi 3.3, 3.4 согласно описанию структур и функций, приведенных в Руководстве программиста Lusbapi.

2. Режимы ЦАП

Модули E14-140(M) с опцией ЦАП обеспечивают возможность вывода аналоговых сигналов по двум независимым каналам. Доступные режимы ЦАП различаются в разных моделях и версиях модулей, однако более поздние версии обратно совместимы с более ранними. Ниже по тексту отличия снабжены соответствующими ремарками.

Каждому режиму ЦАП посвящен отдельный параграф настоящей главы. Здесь приводится их краткий список:

Однократный вывод на ЦАП позволяет по команде от компьютера установить на выходах ЦАП заданное напряжение. При этом отсутствует какая-либо привязка ко времени, поэтому данный режим пригоден только для вывода постоянных (редко изменяющихся) величин.

Потоковый режим ЦАП используется для генерации произвольных сигналов с заданной частотой дискретизации, при этом данные постоянно подкачиваются из компьютера по USB. Сигнал может иметь произвольную форму и длительность, однако на прикладную программу ложится обязанность обеспечивать подкачку данных с достаточной скоростью в течение всего периода генерации.

Циклический режим ЦАП обеспечивает генерацию периодических сигналов (повторяющихся последовательностей). В этом режиме данные загружаются в память модуля один раз перед запуском, подкачка по USB не требуется, но период последовательности ограничен размером буферной памяти модуля.

Расширенные функции E14-140-M (начиная с версии 3.10) позволяют дополнительно:

- Задавать конечную длину выводимой последовательности отсчетов (функция "автостоп").

- Автоматически устанавливать на выходах ЦАП заданную константу при остановке вывода.
- В циклическом режиме начинать вывод с произвольного смещения.
- Осуществлять серии пусков ЦАП в циклическом режиме без перезагрузки данных.

Сводная таблица режимов ЦАП и их доступности в разных версиях модулей:

РЕЖИМ	**E14-140**	**E14-140-M версии < 3.10**	**E14-140-M версии ≥ 3.10**
Однократный (12 бит)	есть	есть (эмуляция)	есть (эмуляция)
Однократный (16 бит)	нет	есть	есть
Потоковый (16 бит)	нет	есть	есть, с расширенными функциями
Циклический (16 бит)	нет	нет	есть, с расширенными функциями

2.1. Форматы данных ЦАП

В модули E14-140-M устанавливается 16-разрядный ЦАП, отсчет которого является знаковым целым числом от –32768 (соответствует напряжению –5 В) до 32767 (соответствует напряжению +5 В). Схемотехника модуля такова, что данные всегда выводятся на оба канала ЦАП, причем обновление выходного напряжения по обоим каналам происходит синхронно. (В частности, на E14-140-M можно осуществить вывод дифференциального сигнала с размахом до 10 В, подавая в каналы ЦАП данные в противофазе).

Таким образом, элементарной единицей информации для ЦАП E14-140-M является "двухканальный отсчет" (иначе говоря, кадр), состоящий из двух 16-битных слов: первое слово содержит отсчет первого канала, второе слово – отсчет второго канала. Двухканальный отсчет можно представить, например, как массив:

SHORT dac_sample[2];

где dac_sample[0] = отсчет канала 1, dac_sample[1] = отсчет канала 2.

Формат слова – little-endian, т.е. младший байт первым.

В более старые модули E14-140 устанавливается 12-разрядный ЦАП, отсчет которого является знаковым целым числом от –2048 (соответствует напряжению –5 В) до 2047 (соответствует напряжению +5 В). При этом вывод данных происходит отдельными командами, по одному каналу за вызов функции, поэтому говорить о кадре данных не имеет смысла.

2.2. Корректировка данных ЦАП

Схемотехника и использованные компоненты обеспечивают линейность передаточной характеристики ЦАП. Однако выходные напряжения ЦАП могут иметь некоторое смещение нуля и неточность коэффициентов передачи, которые можно компенсировать калибровочными коэффициентами, хранящимися в энергонезависимой памяти модуля. В силу ограниченной вычислительной мощности процессора модуль не производит автоматическую корректировку выводимых на ЦАП данных, и задача корректировки возлагается на прикладную программу.

Калибровочные коэффициенты располагаются в полях Dac.OffsetCalibration[] и Dac.ScaleCalibration[] структуры служебной информации MODULE_DESCRIPTION_E140. Эти поля представляют собой массивы из DAC_CALIBR_COEFS_QUANTITY_E140 чисел типа double. Массив Dac.OffsetCalibration[] содержит коэффициенты для корректировки

смещения нуля в каждом канале ЦАП, а массив Dac.ScaleCalibration[] – коэффициенты для корректировки масштаба.

Корректировка данных производится по формуле: Y = (X + A) * B, где X – нескорректированный код отсчета ЦАП, Y – скорректированный код отсчета ЦАП (передается в модуль), A – коэффициент смещения нуля, B – коэффициент масштаба.

Из соображений обратной совместимости коэффициенты смещения нуля Dac.OffsetCalibration[] имеют размерность кода 12-разрядного ЦАП, т.е. при работе с 16-разрядными данными необходимо умножать их на 16.0.

Коэффициенты масштаба безразмерны, поэтому над ними никаких дополнительных действий не требуется.

Например, для первого канала ЦАП в 12-битном режиме:

Y1 = (SHORT)((X1 + Dac.OffsetCalibration[0]) * Dac.ScaleCalibration[0])

Для второго канала ЦАП в 16-битном режиме:

Y2 = (SHORT)((X2 + (Dac.OffsetCalibration[1] * 16.0)) * Dac.ScaleCalibration[1])

2.3. Однократный вывод на ЦАП

Однократный вывод на ЦАП позволяет по команде от компьютера установить на выходах ЦАП заданное напряжение. При этом отсутствует какая-либо привязка ко времени, поэтому данный режим пригоден только для вывода постоянных (редко изменяющихся) величин. Функции однократного вывода выполняются достаточно медленно: время их выполнения определяется процессом обработки команд по USB и является неопределенной величиной порядка десятков миллисекунд.

Функции однократного вывода можно использовать только тогда, когда не запущен потоковый или циклический вывод на ЦАП. В противном случае функция возвратится с ошибкой.

При работе с модулями E14-140-M, имеющими 16-битный ЦАП с синхронным обновлением, для однократного вывода следует использовать функцию *DAC_SAMPLES()*.

При работе с более старыми модулями E14-140, имеющими 12-битный ЦАП с асинхронным обновлением, следует использовать функцию *DAC_SAMPLE()*. В модулях E14-140-M эта функция эмулируется для совместимости со старыми программами, однако использовать ее в программах, создаваемых в расчете на E14-140-M, не следует.

2.4. Параметры потокового и циклического режимов ЦАП

Данная информация относится только к модулю E14-140-M.

Перед запуском ЦАП в потоковом или циклическом режиме необходимо установить параметры, управляющие процессом вывода на ЦАП. Для этого используется структура *DAC_PARS_E140*, общая для потокового и циклического режимов. (На однократный вывод эти параметры не влияют).

Назначение полей структуры DAC_PARS_E140 следующее:

- **double DacRate**: частота дискретизации ЦАП в кГц.

Это частота, с которой выводятся отсчеты ЦАП (в отличие от АЦП, здесь частота не делится на количество каналов, т.е. *оба канала обновляются синхронно* с частотой DacRate кГц.

Допустимые значения частоты дискретизации – от 25 до 200 (кГц). При этом необходимо учитывать, что доступный диапазон частот не непрерывен, а образует сетку частот, рассчитываемую по формуле DacRate = 200 / N кГц, где N – целое число от 1 до 8. При выполнении функции *SET_DAC_PARS()* поле DacRate при необходимости корректируется до ближайшей доступной частоты.

- **BYTE SyncWithADC**: позволяет задать режим синхронизации пуска ЦАП с пуском АЦП.

Если SyncWithADC = 0, то вывод сигнала на ЦАП начинается сразу же после вызова функции *START_DAC()* (при условии наличия данных в буфере).

Если SyncWithADC = 1, то вызовом *START_DAC()* включается режим ожидания пуска АЦП (в соответствии с запрограммированным в АЦП условием пуска), и вывод первого отсчета на ЦАП синхронизируется с началом сбора данных АЦП.

Когда пуск ЦАП синхронизирован с пуском АЦП, в силу схемотехнических особенностей модуля первый отсчет АЦП предшествует по времени появлению первого отсчета на выходах ЦАП.

Иначе говоря, если при SyncWithADC = 1 и одинаковых частотах дискретизации АЦП и ЦАП соединить выход ЦАП со входом АЦП, то первый отсчет АЦП даст напряжение на выходе ЦАП, которое было до пуска, второй отсчет АЦП будет соответствовать первому отсчету ЦАП, третий отсчет АЦП – второму отсчету ЦАП и т.д.

Время между началом преобразования АЦП и обновлением выходов ЦАП стабильно и составляет порядка 200 нс.

Эту особенность следует учитывать при построении на базе E14-140-M замкнутых измерительных установок по схеме "ЦАП → исследуемая цепь → АЦП" (например, измерителей АЧХ/ФЧХ): в таких случаях следует отбрасывать первый отсчет АЦП.

- **BYTE RunMode**: задает режим работы ЦАП.

Это битовое поле, которое задается как логическое "или" (сумма) следующих констант:

DAC_STREAM_MODE_E140	0x00	Выбирает потоковый режим ЦАП.
DAC_LOOP_MODE_E140	0x10	Выбирает циклический режим ЦАП.
DAC_CONST_ON_STOP_E140	0x01	Включает установку константы по окончании вывода на ЦАП.
DAC_NO_CLEAR_BUF_E140	0x02	Отключает очистку буфера при автостопе в циклическом режиме.

DAC_CONST_ON_STOP_E140. Если этот флаг установлен, то при остановке ЦАП в потоковом или в циклическом режиме, независимо от значения последнего выведенного отсчета и от способа остановки (функцией *STOP_DAC()* или по достижению заданного числа отсчетов), на выходах ЦАП устанавливаются напряжения в соответствии с полем StopConst.

В противном случае при остановке ЦАП на выходах сохраняется напряжение, соответствующее последнему выведенному отсчету из потока данных.

DAC_NO_CLEAR_BUF_E140. Этот флаг имеет значение только в циклическом режиме, когда задано конечное число выводимых отсчетов. Если этот флаг установлен, то при автоматической остановке циклического режима ЦАП (по достижению заданного числа отсчетов) **не** производится очистка буфера, и можно сделать повторный запуск функцией *START_DAC()* без необходимости повторной загрузки данных.

Примечание: в модулях E14-140-M с версией встроенного ПО ниже 3.10 поле RunMode может принимать только значения 0 или 1 (поддерживается только потоковый режим), а

значение стоповой константы всегда нулевое (поле StopConst игнорируется). В версии lusbapi 3.3 поле RunMode называлось соответственно SetZeroOnStop.

- **DWORD TotalSamples**: количество выводимых отсчетов.

  Если TotalSamples = 0, то ЦАП работает до остановки функцией *STOP_DAC()*.

  Если же TotalSamples > 0, то включается режим "автостоп", в котором модуль автоматически остановит вывод на ЦАП после того, как будет выведено ровно TotalSamples двухканальных отсчетов (кадров) ЦАП.

  ! Количество **двухканальных** отсчетов равно **половине** количества 16-битных слов в потоке выводимых данных (по одному на каждый канал) или, что то же самое, ¼ размера выводимых данных в байтах.

  Внимание! Передача данных для ЦАП в модуль должна выполняться блоками, кратными 256 байтам (64 двухканальных отсчета).

  В любом случае вызовом функции *STOP_DAC()* можно остановить вывод на ЦАП досрочно.

  В циклическом режиме TotalSamples может принимать любые значения (0 – непрерывный вывод, 1 – вывод одного отсчета, и т.д. до 2^{32}-1 отсчетов. TotalSamples может быть не кратно периоду цикла (PreloadSamples, см. ниже).

  В потоковом режиме значение TotalSamples (если оно не равно нулю) не может быть меньше PreloadSamples.

  Хотя передача данных для ЦАП в модуль функцией WriteData() по условию должна выполняться блоками, кратными 256 байтам (64 двухканальных отсчета), значение TotalSamples не обязано быть кратным 64. Если количество полезных отсчетов, загружаемых в модуль, не кратно 64, необходимо дополнить последнюю порцию данных произвольными отсчетами, которые останутся в буфере и не попадут на выход ЦАП.

  Примечание: в E14-140-M с версией встроенного ПО ниже 3.10 это поле отсутствует (функции "автостоп" нет).

- **WORD PreloadSamples**: количество отсчетов, загружаемых перед пуском ЦАП.

  Это поле имеет несколько различный смысл в потоковом и в циклическом режимах.

  а) В циклическом режиме параметр PreloadSamples определяет период выводимой последовательности, т.е. полное количество двухканальных отсчетов, загружаемых в буфер ЦАП.

  Диапазон значений PreloadSamples в циклическом режиме – от 1 до 5120, либо 0 (используется значение по умолчанию, равное 2048).

  б) В потоковом режиме ЦАП параметр PreloadSamples задает длину предварительной загрузки, т.е. минимальное количество двухканальных отсчетов в буфере ЦАП, при котором разрешен пуск. Смысл этого параметра в том, чтобы предотвратить опустошение буфера в начале работы, не начиная потоковый вывод на ЦАП, пока по USB не будет подкачана первая порция данных. (например, если функция *START_DAC()* вызвана раньше, чем WriteData()).

  Диапазон значений PreloadSamples в потоковом режиме – от 128 до 5120, либо 0 (используется значение по умолчанию, равное 2048).

Количество **двухканальных** отсчетов равно **половине** количества 16-битных слов в потоке выводимых данных (по одному на каждый канал) или, что то же самое, ¼ размера выводимых данных в байтах.

Внимание! Передача данных для ЦАП в модуль должна выполняться блоками, кратными 256 байтам (64 двухканальных отсчета).

Хотя передача данных для ЦАП в модуль функцией WriteData() по условию должна выполняться блоками, кратными 256 байтам (64 двухканальных отсчета), значение PreloadSamples не обязано быть кратным 64. Если количество полезных отсчетов, загружаемых в модуль, не кратно 64, необходимо дополнить последнюю порцию данных произвольными отсчетами, которые останутся в буфере и не попадут на выход ЦАП.

Примечание: в E14-140-M с версией встроенного ПО ниже 3.10 это поле отсутствует (длина предварительной загрузки для потокового режима всегда равна 2048 двухканальным отсчетам).

- **SHORT StopConst[2]**: "стоповая константа" для установки при завершении потокового или циклического режима ЦАП.

Это поле имеет значение только в том случае, если в поле RunMode установлен флаг DAC_CONST_ON_STOP_E140. Оно содержит отсчеты для первого и второго каналов ЦАП (соответственно StopConst[0] и StopConst[1]), которые должны быть выведены после остановки ЦАП (функцией *STOP_DAC()* или автоматически после TotalSamples отсчетов, если используется режим "автостоп").

Если флаг DAC_CONST_ON_STOP_E140 в RunMode не установлен, то стоповая константа не используется, и при остановке на ЦАП сохраняется напряжение, соответствующее последнему выведенному отсчету из потока данных.

Следует заметить, что если стоповая константа выводится в результате вызова функции *STOP_DAC()*, то в силу асинхронности процессов выполнения команд от компьютера и вывода на ЦАП возможна ситуация, когда в течение короткого времени после остановки (порядка нескольких десятков мкс) на выходах ЦАП сохраняется напряжение, соответствующее последнему выведенному отсчету из потока данных, а потом устанавливается стоповая константа.

В режиме "автостоп" модуль добавляет стоповую константу непосредственно в поток данных, и она выводится без задержки как (TotalSamples+1)-й отсчет.

Примечание: в E14-140-M с версией встроенного ПО ниже 3.10 это поле отсутствует (значение стоповой константы всегда нулевое).

- **WORD StartOffset**:начальное смещение при работе в циклическом режиме.

Это поле задает смещение, с которого начинается вывод на ЦАП в циклическом режиме (иначе говоря, количество двухканальных отсчетов, которые пропускаются в первом периоде). StartOffset может принимать значения от 0 (вывод сигнала начинается с первого отсчета, загруженного в буфер) до PreloadSamples - 1, поскольку поле PreloadSamples (см. выше) в циклическом режиме задает период генерируемой последовательности.

При выводе периодических сигналов (например, отсчетов синусоиды) поле StartOffset удобно использовать для задания фазы. Также, комбинируя значения StartOffset и TotalSamples, можно вывести на ЦАП любой фрагмент загруженных данных – например, загрузить в буфер набор коротких последовательностей (импульсов разной формы) и выводить их на ЦАП в произвольном порядке по команде от компьютера. При этом, если использовать флаг DAC_NO_CLEAR_BUF_E140 поля RunMode (не очищать буфер ЦАП при остановке вывода), то можно обойтись одной загрузкой данных для целой серии пусков.

Примечание: в E14-140-M с версией встроенного ПО ниже 3.10 это поле отсутствует (нет циклического режима).

2.5. Порядок работы с ЦАП в потоковом режиме

Данная информация относится только к модулю E14-140-M.

Предполагается, что соединение с модулем установлено и прочитан дескриптор устройства. Тогда для работы с ЦАП в потоковом режиме (с постоянной подкачкой данных по USB) необходимо выполнить следующую последовательность действий:

1) Сбросить состояние ЦАП и очистить буфер, вызвав функцию *STOP_DAC()*.

2) Заполнить структуру *DAC_PARS_E140* требуемыми значениями параметров и передать параметры в модуль функцией *SET_DAC_PARS()*.

3) (*шаги 3 и 4 можно выполнять в любом порядке*) Начать загрузку данных в модуль функцией WriteData() или системным вызовом WriteFile() (рекомендуется передавать данные в отдельном потоке, используя режим Overlapped I/O и двойную буферизацию). Передача данных должна выполняться блоками, кратными 256 байтам (128 16-битным словам, 64 двухканальным отсчетам).

4) Подать команду пуска ЦАП функцией *START_DAC()*. При этом в слове состояния модуля (см. описание функции *GET_MODULE_STATE()*) устанавливается флаг DEVSTATE_DAC_STARTED, но фактически вывод на ЦАП начинается только при наличии в памяти модуля не менее DacPars.PreloadSamples двухканальных отсчетов (а при использовании синхронизации ЦАП с АЦП модуль будет ждать также пуска АЦП).

5) В течение всего периода генерации сигнала необходимо обеспечивать подкачку данных с достаточной скоростью во избежание опустошения буфера и возникновения "дыр" в сигнале (которые модуль заполняет нулевыми отсчетами). Для этого нужно, чтобы в любой момент времени в системе был как минимум один активный (не завершившийся) вызов в режиме Overlapped. Например, объявляется два буфера ("четный" и "нечетный"), и в начале работы в очередь ставятся сразу два запроса на запись в модуль. Потом, как только завершается одна передача (что отслеживается по событию hEvent структуры OVERLAPPED) – а в это время операционная система уже начала передавать второй буфер – следует сразу же подготовить и поставить в очередь новую порцию данных.

6) Если количество отсчетов ограничено параметром DacPars.TotalSamples, то по окончании вывода модуль автоматически выполнит остановку ЦАП, о чем свидетельствует сброс флага DEVSTATE_DAC_STARTED в слове состояния модуля. При выводе неопределенного количества отсчетов (DacPars.TotalSamples = 0), либо досрочно, вывод на ЦАП можно прекратить функцией *STOP_DAC()*, отменив незавершенные асинхронные запросы на передачу данных при помощи системного вызова CancelIo().

Параметры ЦАП и форматы вызова функций библиотеки подробно описаны в соответствующих разделах.

2.6. Порядок работы с ЦАП в циклическом режиме

Данная информация относится только к модулю E14-140-M.

Предполагается, что соединение с модулем установлено и прочитан дескриптор устройства. Тогда для работы с ЦАП в циклическом режиме (генерация периодических или

коротких последовательностей с однократной загрузкой данных) необходимо выполнить следующую последовательность действий:

1) Сбросить состояние ЦАП и очистить буфер, вызвав функцию *STOP_DAC()*.

2) Заполнить структуру *DAC_PARS_E140* требуемыми значениями параметров и передать параметры в модуль функцией *SET_DAC_PARS()*. При этом параметр DacPars.PreloadSamples устанавливается равным количеству двухканальных отсчетов в периоде сигнала (от 1 до 5120).

3) Загрузить в модуль все данные, содержащие DacPars.PreloadSamples двухканальных отсчетов сигнала, функцией WriteData() или системным вызовом WriteFile(). Передача данных должна выполняться блоками, кратными 256 байтам (128 16-битным словам, 64 двухканальным отсчетам), поэтому размер данных (в байтах) должен делиться на 256. При необходимости данные надо дополнить произвольными (незначащими) отсчетами. Например:

```
size_t NumWords = (DacPars.PreloadSamples * 2 + 127) & ~127;
SHORT* buf = new SHORT[NumWords];
```

(здесь элементы buf[] с четными индексами соответствуют первому каналу, а с нечетными – второму).

4) Подать команду пуска ЦАП функцией *START_DAC()*. При этом в слове состояния модуля (см. описание функции *GET_MODULE_STATE()*) устанавливается флаг DEVSTATE_DAC_STARTED и, поскольку в память модуля уже загружены все данные, начинается вывод на ЦАП (а при использовании синхронизации ЦАП с АЦП модуль будет сначала ждать пуска АЦП).

5) В течение периода генерации сигнала модуль работает автономно. При желании можно периодически обращаться к сервисным функциям, например, *GET_MODULE_STATE()*.

6) Если количество отсчетов ограничено параметром DacPars.TotalSamples, то по окончании вывода модуль автоматически выполнит остановку ЦАП, о чем свидетельствует сброс флага DEVSTATE_DAC_STARTED в слове состояния модуля. При выводе неопределенного количества отсчетов (DacPars.TotalSamples = 0), либо досрочно, вывод на ЦАП можно прекратить функцией *STOP_DAC()*.

Примечание: загрузку данных (шаг 3) можно выполнять один раз для серии пусков в циклическом режиме, если использовать флаг DAC_NO_CLEAR_BUF_E140 поля RunMode для подавления очистки буфера при автоматической остановке, а при ручной остановке – вызывать функцию *STOP_DAC()* с параметром DAC_NO_CLEAR_BUF_E140.

Параметры ЦАП и форматы вызова функций библиотеки подробно описаны в соответствующих разделах.

3. Структуры для работы с ЦАП

3.1. Структура DAC_PARS_E140

Структура DAC_PARS_E140 описана в файле `\DLL\Include\Lusbapi.h` и представлена ниже:

```
struct DAC_PARS_E140
{
    BYTE SyncWithADC;           // признак синхронизации с пуском АЦП
    BYTE RunMode;               // флаги режимов работы
    double DacRate;             // частота дискретизации ЦАП, кГц
    DWORD TotalSamples;         // количество выводимых отсчетов
    WORD PreloadSamples;        // количество отсчетов, загружаемых до пуска
    SHORT StopConst[2];         // константа, выводимая при остановке
    WORD StartOffset;           // начальное смещение в циклическом режиме
};
```

Назначение полей структуры подробно описано в разделе "*Параметры потокового и циклического режимов ЦАП*".

Перед началом работы с ЦАП в потоковом или циклическом режиме необходимо заполнить поля данной структуры и передать ее в модуль с помощью функции *SET_DAC_PARS()*. Также при необходимости можно считать из модуля текущие параметры ЦАП, используя функцию *GET_DAC_PARS()*.

В версии Lusbapi 3.3 данная структура содержит меньше полей. При необходимости использования новых возможностей следует определить вспомогательный тип:

```
#pragma pack(1)
struct DAC_PARS_E140_INT
{
    WORD RateDiv;                   // 0…7, равно (200 / DacRate) – 1
    BYTE SyncWithADC;
    BYTE RunMode;
    DWORD TotalSamples;
    WORD PreloadSamples;
    SHORT StopConst[2];
    WORD StartOffset;
};
```

и использовать недокументированные вызовы GetArray() и PutArray():

```
DAC_PARS_E140_INT my_dac_pars;
pModule->GetArray((BYTE*)&my_dac_pars, sizeof(my_dac_pars), 0x0160);
pModule->PutArray((BYTE*)&my_dac_pars, sizeof(my_dac_pars), 0x0160);
```

3.2. Структура MODULE_STATE_E140

Структура MODULE_STATE_E140 описана в файле `\DLL\Include\Lusbapi.h` и представлена ниже:

```
struct MODULE_STATE_E140
{
    DWORD DeviceState;          // слово состояния модуля (флаги)
    DWORD ADC_OverrunCount;     // счетчик ошибок переполнения буфера АЦП
    DWORD DAC_UnderrunCount;    // счетчик ошибок опустошения буфера ЦАП
    DWORD ADC_SampleCount;      // счетчик отсчетов, полученных от АЦП
    DWORD DAC_SampleCount;      // счетчик отсчетов, выведенных на ЦАП
};
```

Получение информации о состоянии модуля производится с помощью функции *GET_MODULE_STATE()*. Назначение полей данной структуры подробно описано в пояснении к этой функции.

В версии Lusbapi 3.3 данная структура отсутствует. Ее можно определить самостоятельно и использовать недокументированный вызов GetArray():

```
MODULE_STATE_E140 my_state;
pModule->GetArray((BYTE*)&my_state, sizeof(my_state), 0x10C0);
```

4. Функции для работы с ЦАП

4.1. Однократный вывод на ЦАП, 12 бит. Для E14-140 (не "-M").

Формат: **BOOL** ***DAC_SAMPLE***(*SHORT * const DacData, WORD DacChannel*)

Назначение:

Устанавливает на канале ЦАП с номером DacChannel заданное напряжение в соответствии с 12-битным значением *DacData. Величина *DacData должна находиться в диапазоне от –2048 до 2047, иначе функция автоматически ограничит значение указанными пределами. О соответствии кода ЦАП величине устанавливаемого на выходе напряжения см. *Форматы данных ЦАП*.

Эту функцию следует использовать только в программах, рассчитанных на модуль E14-140 (не E14-140-M). В модулях E14-140-M эмулируется для совместимости (при этом обновляются все равно оба канала, а для неактивного канала берется значение, сохраненное из предыдущих вызовов функций однократного вывода).

Передаваемые параметры:

- DacData – указатель на устанавливаемое значение в кодах ЦАП (от –2048 до 2047);
- DacChannel – номер канала ЦАП (0 или 1).

Возвращаемое значение: *TRUE* – функция успешно выполнена;
FALSE – при выполнении произошла ошибка.

4.2. Однократный вывод на ЦАП, 16 бит (для E14-140-M)

Формат: BOOL ***DAC_SAMPLES***(*SHORT * const DacData1, SHORT * const DacData2*)
Назначение: Устанавливает на каналах ЦАП заданное напряжение в соответствии с 16-битными значениями *DacData1 (первый канал) и *DacData2 (второй канал). О соответствии кода ЦАП величине устанавливаемого на выходе напряжения см. *Форматы данных ЦАП*.
Передаваемые параметры: • DacData1 – указатель на устанавливаемое значение для первого канала ЦАП; • DacData2 – указатель на устанавливаемое значение для второго канала ЦАП.
Возвращаемое значение: *TRUE* – функция успешно выполнена; *FALSE* – при выполнении произошла ошибка.

4.3. Установка параметров ЦАП

Формат: BOOL ***SET_DAC_PARS***(*DAC_PARS_E140 * const DacPars*)
Назначение: Передает в модуль E14-140-M параметры для потокового или циклического режима ЦАП, определяемые структурой *DacPars. Установленные параметры применяются только при следующем обращении к функции *START_DAC()*. Если процесс вывода на ЦАП уже запущен, то изменения, вносимые данной функцией, не вступят в силу до следующего пуска. Рекомендуется изменять параметры ЦАП при остановленном выводе (после *STOP_DAC()*). Назначение полей структуры *DAC_PARS_E140* описано в разделе "*Параметры потокового и циклического режимов ЦАП*". Если параметры ЦАП некорректны (например, не выполняются требования к диапазонам значений параметров для выбранного режима работы), функция может завершиться с ошибкой.
Примечание: Чтобы использовать новые поля структуры с версией Lusbapi 3.3, данную функцию можно заменить вызовом PutArray((BYTE*)&pars, sizeof(DAC_PARS_E140_INT), 0x0160), где pars – структура типа *DAC_PARS_E140_INT*.
Передаваемые параметры: • DacPars – указатель на структуру типа *DAC_PARS_E140*, содержащую требуемые параметры ЦАП.
Возвращаемое значение: *TRUE* – функция успешно выполнена; *FALSE* – при выполнении произошла ошибка.

4.4. Получение текущих параметров ЦАП

Формат: **BOOL** ***GET_DAC_PARS***(*DAC_PARS_E140 * const DacPars*)

Назначение:

Считывает из модуля E14-140-M параметры для потокового или циклического режима ЦАП и заполняет ими структуру *DacPars.

Назначение полей структуры *DAC_PARS_E140* описано в разделе "*Параметры потокового и циклического режимов ЦАП*".

Примечание: Чтобы использовать новые поля структуры с версией Lusbapi 3.3, данную функцию можно заменить вызовом GetArray((BYTE*)&pars, sizeof(DAC_PARS_E140_INT), 0x0160), где pars – структура типа *DAC_PARS_E140_INT*.

Передаваемые параметры:

- DacPars – указатель на структуру типа *DAC_PARS_E140*, в которую записываются текущие параметры ЦАП.

Возвращаемое значение: *TRUE* – функция успешно выполнена;
FALSE – при выполнении произошла ошибка.

4.5. Запуск потокового или циклического вывода на ЦАП

Формат: **BOOL** ***START_DAC***(*void*)

Назначение:

Запускает вывод на ЦАП модуля E14-140-M в потоковом или циклическом режиме.

Параметры работы ЦАП должны быть предварительно переданы в модуль с помощью функции *SET_DAC_PARS()*. Загрузку данных в модуль можно осуществлять функцией WriteData().

Запуск возможен только из состояния "ЦАП остановлен", поэтому перед запуском, если состояние модуля неизвестно, рекомендуется выполнять функцию *STOP_DAC()*.

Примечание:

При успешном выполнении данная функция устанавливает в слове состояния модуля флаг DEVSTATE_DAC_STARTED, очищает счетчик опустошений буфера и инициирует пуск ЦАП. При этом фактическая выдача сигнала на ЦАП может начаться не сразу, если задан режим синхронизации пуска ЦАП с пуском с АЦП, либо если на момент получения команды пуска в буфере модуля недостаточно данных (меньше, чем задано параметром PreloadSamples в структуре *DAC_PARS_E140*).

Передаваемые параметры: нет

Возвращаемое значение: *TRUE* – функция успешно выполнена;
FALSE – при выполнении произошла ошибка.

4.6. Останов потокового или циклического вывода на ЦАП

Формат: BOOL ***STOP_DAC*****(***BYTE NoClearBuf = 0***)**
Назначение: Останавливает вывод на ЦАП модуля E14-140-M в потоковом или циклическом режиме, если он был запущен. Очищает буфер, хранящий отсчеты данных для вывода на ЦАП (если параметром функции не указано иначе). Сбрасывает в слове состояния модуля флаг DEVSTATE_DAC_STARTED. Если ЦАП был остановлен или находился в ожидании пуска, функция сбрасывает состояние ЦАП и завершается без ошибки. Поэтому данную функцию можно (и рекомендуется) использовать для очистки и приведения ЦАП в исходное состояние (например, в начале работы программы).
Примечание: в версии Lusbapi 3.3 параметр NoClearBuf отсутствует.
Передаваемые параметры: • *NoClearBuf* (необязательный) – подавляет очистку буфера ЦАП, если его значение равно DAC_NO_CLEAR_BUF_E140.
Возвращаемое значение: *TRUE* – функция успешно выполнена; *FALSE* – при выполнении произошла ошибка.

4.7. Передача массива данных в ЦАП

См. описание функции WriteData() в Руководстве программиста Lusbapi 3.3 со следующим исправлением: параметр NumberOfWordsToPass должен быть кратен 128, а не 256.

4.8. Чтение информации о состоянии модуля

Формат: BOOL *GET_MODULE_STATE*(*MODULE_STATE_E140 * const State*)

Назначение:

Считывает из модуля E14-140-M (версия встроенного ПО не ниже 3.10) структуру с информацией о состоянии устройства.

Назначение полей структуры *MODULE_STATE_E140*:

- **DWORD DeviceState** – слово состояния модуля, в котором могут быть установлены следующие флаги (биты):

DEVSTATE_ADC_STARTED (0x01): запущен сбор АЦП. Устанавливается вызовом START_ADC(), сбрасывается вызовом STOP_ADC(). Флаг отражает логическое состояние модуля, поэтому он установлен, даже если АЦП только ожидает пуска.

DEVSTATE_DAC_STARTED (0x02): запущен вывод на ЦАП. Устанавливается вызовом *START_DAC()*, сбрасывается вызовом *STOP_DAC()* или при автоматической остановке ЦАП (в режиме вывода фиксированного количества отсчетов). Флаг отражает логическое состояние модуля, поэтому он установлен, даже если ЦАП только ожидает пуска.

- **DWORD ADC_OverrunCount** – счетчик ошибок переполнения буфера АЦП. Такие ошибки возникают, если программа не успевает забирать данные с достаточной скоростью в потоковом режиме АЦП. Счетчик увеличивается на 1 каждый раз, когда модулю приходится отбросить блок собранных данных (32 16-битных отсчета АЦП). При вызове START_ADC() счетчик обнуляется.
- **DWORD DAC_UnderrunCount** – счетчик ошибок опустошения буфера ЦАП. Такие ошибки возникают, если программа не успевает записывать данные с достаточной скоростью в потоковом режиме ЦАП. Счетчик увеличивается на 1 каждый раз, когда модулю приходится вставить вместо отсутствующих данных блок из 64 нулевых двухканальных отсчетов ЦАП. При вызове *START_DAC()* счетчик обнуляется.
- **DWORD ADC_SampleCount** – счетчик отсчетов, полученных с АЦП в потоковом режиме. Обнуляется при вызове START_ADC(), затем увеличивается порциями по 32 отсчета по мере сбора данных. Не включает отброшенные данные, если были ошибки переполнения буфера АЦП.
- **DWORD DAC_SampleCount** – счетчик двухканальных отсчетов (кадров), выведенных на ЦАП в потоковом или циклическом режиме. Обнуляется при вызове *START_DAC()*, затем увеличивается по мере фактического вывода сигнала на ЦАП. Не включает длину "дыр" в сигнале, если были ошибки опустошения буфера ЦАП.

Примечание: в Lusbapi 3.3 данную функцию можно заменить вызовом GetArray((BYTE*)State, sizeof(MODULE_STATE_E140), 0x10C0).

Передаваемые параметры:

- State – указатель на структуру типа *MODULE_STATE_E140*, в которую записывается состояние модуля.

Возвращаемое значение: *TRUE* – функция успешно выполнена;
FALSE – при выполнении произошла ошибка